

End-to-End Wireless Digital Communication System of FPGA Based Software Defined Radio

Tuan M. Dang, Thomas Gonnot and Jafar Saniie
*Department of Electrical and Computer Engineering
 Illinois Institute of Technology, Chicago, Illinois, USA*

Abstract— This paper describes the design and implementation of an End-to-End digital wireless communication system (E2E) system on the Zynq-7000 Field Programmable Gate Array (FPGA) and Evaluation Development (Zed) Board with an Analog Devices AD9364 Wideband Transceiver from a model-based design system. Furthermore, the paper presents the design path to reconfigure the software controlled logic blocks for additional functionality such that the E2E system could be rapidly redeployed. The results will show how to realize the E2E system and the BER vs. distance plot will verify the theoretical expectation between uncoded and coded modulation for over the air transmission and reception.

I. INTRODUCTION

In this paper, we introduce Field Programmable Gate Arrays (FPGAs), an array of programmable logic blocks, as a platform for implementing the Software Defined Radio (SDR). The FPGA based SDR is a programmable System-on-a-chip (SoC) that controls the interconnect logic blocks to realize a digital wireless communication system. By using FPGA based SDR, a digital transmitter and receiver, of any communication protocols, can be programmed on the FPGA to be executed in real-time, the same programming code can be reused and executed on another FPGA platform, and rapid reconfiguration of the communication system can be done by changing the FPGA software.

To realize the three points listed above, this paper will detail the design and implementation of a simple end-to-end digital wireless communication system on the FPGA based SDR (E2E). The E2E system consist of two FPGA based SDR used to communicate wirelessly with each other. Each FPGA based SDR consists a Xilinx Zynq-7000 Evaluation and Development Board (ZedBoard) with a front-end Analog Devices (AD) 9364 FMCOMMS4-EBZ RF Transceiver. The choice for using Xilinx FPGA based radio is the availability and accessibility of the FPGA and AD programming libraries, allowing for rapid development and customization of the communication system. Other available platform includes Ettus Research LLC packaged Universal Software Radio Peripherals (USRP) [1].

The design and implementation of the E2E can be broken up into software and hardware partitions. The software partition is implemented on the ARM Processing System (PS),

composed of two ARM Cortex-A9 CPUs. The PS is responsible for data encoding, decoding, and low processing rate tasks such as arithmetic operations [2].

The hardware partition is called the Programmable Logic (PL), which consists the FPGA programmable logic blocks. Here is where custom digital transmission and reception algorithms are implemented. The transmitted and received baseband signals are sent and received from the connected front-end AD9364 Transceiver.

Hardware Description Language (HDL) is used to configure the PL, which implements the digital communication functionalities in hardware [3]. For the SDR transmitter, hardware implementation includes data modulation, transmission filtering, and sending the baseband signal to the front-end of the RF transmitter. For the SDR receiver, the FPGA implementation includes signal acquisition at the RF receiver front-end, Automatic Gain Control (AGC), receiving filter, coarse frequency compensation, fine frequency compensation, timing recovery, and data demodulation [2] [4].

Simulink is the chosen platform to design and implement the E2E system because it brings together different third party software and hardware tools into HDL code for the FPGA. Simulink operates as a behavioral modeling environment, allowing for easy traceability of a signal from end to end. In this modeling environment, each building block (whether the front-end SDR or FPGA components) has been encapsulated by Simulink as input-output abstract function blocks and when connected, can be deployed in real-time simulation or externally on the FPGA device.

II. E2E DESIGN MODEL

A top-level Simulink model for the E2E system is shown in Figure 1. The Quadrature Phase Shift Keying (QPSK) modulation and demodulation are used for the Transmitter and Receiver Logic. QPSK modulation scheme is used for high data signal by pairing successive data bit stream and mapping the pair bits onto one of the four symbol positions in the IQ complex plane. Thus, each symbol is represented by a bit-pair and the overall QPSK efficiently contains twice the data with the same transmit bandwidth as compared to Binary PSK [5].

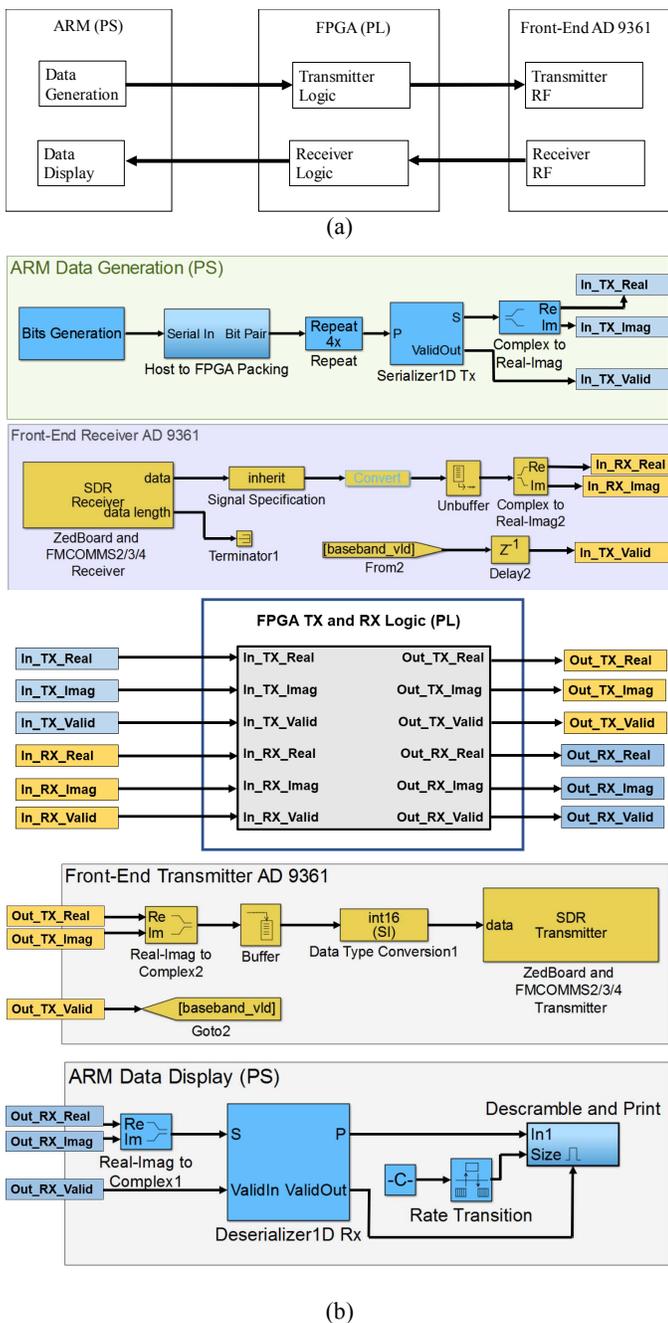


Figure 1. (a) Top level view of end-to-end digital wireless communication system, (b) detailed components of the top level

In the transmission chain, the signal is generated by the ARM (PS) 'Data Generation' block, where a predetermined ASCII message has been digitized and embedded in a predetermined 200-bit payload, with 174 bits reserved for the ASCII message and the remaining 26-bit header is a Unipolar Barker Code for payload identification used for demodulation at the receiver side. Furthermore, an arbitrary 7 bits per ASCII integer is defined, yielding a maximum of 24 character-string that can be sent at each time frame.

Next, the payload is serialized and passed through a 'Serialize bit pairs' block where successive bits are grouped in

pairs and are fed to the 'QPSK modulator: $\frac{\pi}{4}$ ' block. Here, the QPSK modulator has a phase offset of $\pi/4 \approx 0.7071$, and where each pair bits are mapped onto one of the four distinct symbol positions in the IQ complex plane. To prevent Inter-Symbol Interference (ISI), these symbols are passed through the 'FIR filter' block, which is a FIR Low-Pass-Filter (LPF) used to band-limit high frequency symbols [5].

The complete transmit model inside the 'Transmitter Logic' implemented in FPGA using Simulink is shown in Figure 2. Figure 2 also shows the output of the 'Transmitter Logic' being further processed and packaged to a compatible 16-bit signed frame based integers prior to entering the front end of the 'Transmitter RF'. At the 'Transmitter RF', the payload is converted to an analog signal and mixed with the carrier frequency prior to over the air transmission.

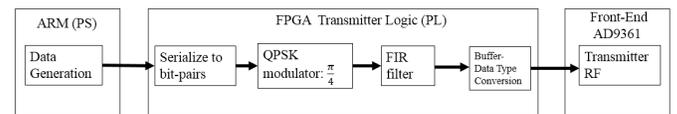


Figure 2. Transmit chain from PS to front-end AD9364

In the receiving chain, the FPGA is used to implement a digital receiver. The incoming analog signal is received at the front-end 'Receiver RF', where it is converted to 16-bits signed complex integers. Figure 3 shows the design of a digital receiver inside the 'Receiver Logic' that will process the incoming discrete samples. Prior to being demodulated by the QPSK demodulator, the input signal underwent frequency, phase, and time corrections to render a modulated QPSK signal.

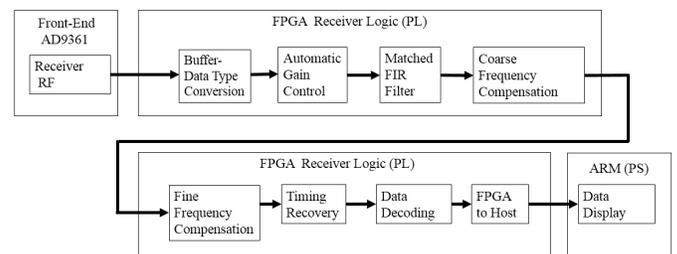


Figure 3. Receive chain from PS to front-end AD9364

The seven stages of the digital receiver are Automatic Gain Control (AGC), matching receiver filter, coarse frequency compensation, fine frequency compensation, timing recovery, data decoding, and FPGA to Host. A description of the seven stages in the 'Receiver Logic' [4] [6]:

1. AGC – amplifies the received signal for decoding
2. Receiver filter – matched filter with transmit FIR filter used to synchronize sampling times.
3. Coarse frequency compensation – finds and add frequency offset to match receiver carrier frequency to the down conversion frequency to baseband. The 'Scrambler' and 'Descrambler' inserted in the 'Transmitter Logic' and 'Receiver Logic' are used to help estimate the frequency compensation.

4. Fine frequency compensation – finds and add phase offset to the received data to match the transmitter’s.
5. Timing Recovery – implements Phase Locked Loop for symbol time synchronization. By using the receiver filter to remove noise and extract the received data periodicity to estimate the sampling period and hence estimate the transmitter signal’s sampling time.
6. Data Decoding – demodulates QPSK input signal using QPSK demodulator.
7. FPGA to Host – converts demodulated QPSK bit stream into 200-bit payload for further decoding by the ARM PS Data Display.

Figure 4 shows the effects on the received signal as it goes through successive demodulating stages, ultimately ending with the constellation de-mapper with each bit pairs de-mapped onto their respective symbol locations $n\pi/4$, $n=1,3,5,7$, to create four clusters. The ‘After Timing Recovery’ shows the final constellation diagram for the QPSK modulator with phase offset $\pi/4$ and Gray Coding. The QPSK demodulator will assemble these bit-pair chronologically to reconstruct the 200-bit payloads to be decoded further by the ARM PS ‘Data Display’ [4] [5] [6].

For the E2E, most of the signal processing such as RF encoding and decoding, and transmit-receive frequency, phase, and time synchronizations are all done in the FPGA. To implement the E2E hardware model, the software interface directly controls the different E2E hardware components for realizing specific applications [4]. For example, post RF processing by the FPGA, the software interface model will translate the 200-bit payload to ASCII characters and print them to a console window.

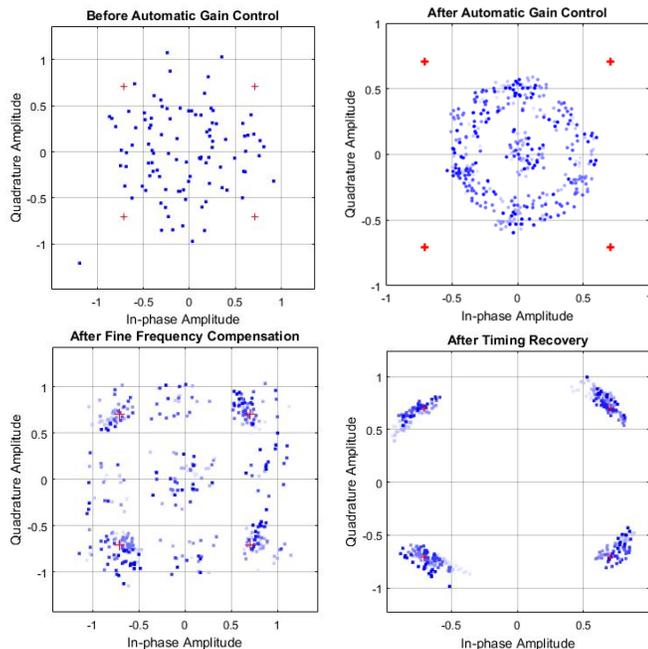


Figure 4. Successive snapshots of the input signal from the front-end of the SDR receiver to the timing recovery stage prior

III. E2E IMPLEMENTATION FROM DESIGN MODEL

The complete E2E design model in Figure 1 can be converted to HDL and implemented in the FPGA using the Simulink’s HDL Workflow. HDL Workflow streamlines the process of translating the high-level design model into HDL, which will be encapsulated as a bitstream file and downloaded onto the FPGA based SDR hardware for deployment. Figure 5 shows the design and implementation of the E2E using the HDL Workflow [2] [5].

HDL Workflow also auto generates the software interface SDR transmit and receive blocks from the Simulink design model in Figure 1. For E2E application, the software interface shown in Figure 6 is used to deploy Frequency Division Duplex (FDD). For two SDRs, A and B, deploying FDD would set the TX_A = 951 MHz, RX_A = 851 MHz, TX_B = 851 MHz and RX_B = 951 MHz. These frequencies are configured in the software interface model and supported by the AD9364 FCOMMS4-EBZ with transmit tunable frequency range 47 MHz to 6 GHz and receive tunable frequency range 70 MHz to 6 GHz. Also, the software interface model is used to decode ASCII symbols and print them to the console window [7].

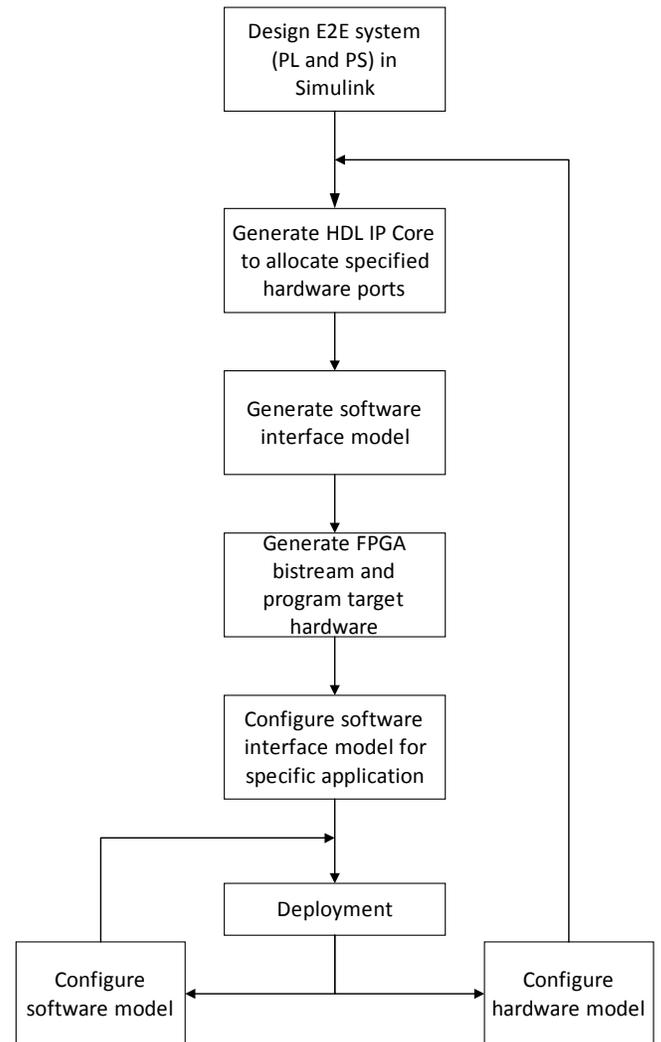
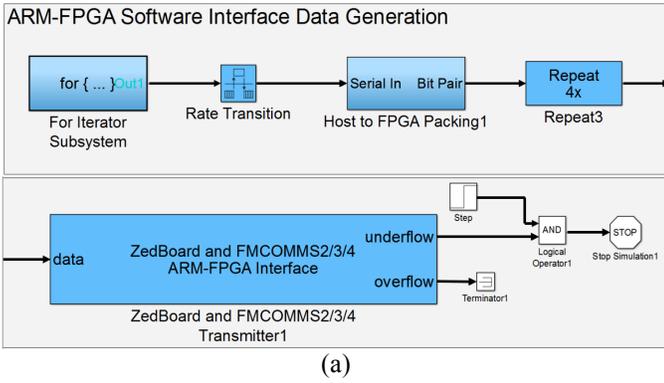
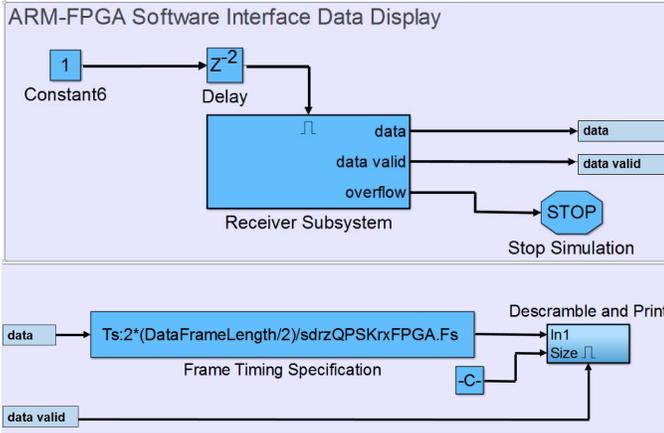


Figure 5. Workflow diagram for designing and implementing E2E



(a)



(b)

Figure 6. (a) ARM-FPGA Software Interface Data Generation (b) ARM-FPGA Software Interface Data Display

IV. APPLICATIONS

Further enhancements can be made to the current E2E system to minimize BER by including Forward Error Correction (FEC) using Convolutional Encoder/Decoder. The current E2E System can be modified by adding Simulink's Convolutional Encoder in the ARM-FPGA Software Interface Data Generation block of the and Convolutional Decoder in the ARM-FPGA Software Interface Data Display (refer to Figure 6).

On the transmitter side, the Convolutional Encoder is placed specifically in front of the "Message Source". Message bits are passed through the Convolutional Encoder that contains a window of predetermined polynomial coefficients. The output is a parity message consists of message bits sliding through the window resulting a sum of product of each informational bit and polynomial coefficients. The constraint length containing the polynomial coefficients is the size of the sliding window. Equation 2 describes the convolutional encoder process [8].

$$p_i[n] = \sum_{j=0}^{k-1} g_i[j]x[n-j] \quad (2)$$

FEC with a rate of $r = 2 \rightarrow \frac{1}{r} = 2$ is specifically used in this paper as a comparison to uncoded QPSK modulator.

Thus, using Equation 2, let the constraint length, $K = 7$ with polynomial coefficients:

$$g_1 = 1111001_2(171_8) \rightarrow p_1[n] = \sum_{j=0}^6 g_1[j]x[n-j] \quad (3)$$

$$g_2 = 1011011_2(133_8) \rightarrow p_2[n] = \sum_{j=0}^6 g_2[j]x[n-j] \quad (4)$$

Where $x[n]$ is the input message bits. Furthermore, what is being sent are $p_i[n]$, $i = 1, 2$ since $r = 2$, there are two parity equations. For the uncoded E2E system, a predetermined 174 message bits are allocated. With FEC rate 1/2 then in order for the size of the allocated message bits to stay 174 bits, the original ASCII text must be reduced to 87 bits.

On the receiver side, the Viterbi Decoder is placed specifically in the "Descrambling and Print" block of Figure 6 to decode the incoming parity bits. The Viterbi Decoder uses the Trellis or state diagram to decode the received parity bits. Specifically for this E2E system with FEC rate of 1/2, the received bitstream are paired and determined whether each parity corresponds to a valid state transition in the Trellis diagram. If no errors are found in the received parity bits, then the Viterbi decoder will follow the state transition path in decoding the message. On the other hand, if there is an error in the parity bits, then the Viterbi decoder will find the minimum Hamming distance between the error parity bits with legal parity bits in the Trellis diagram and picked the path with the minimum Hamming distance. This process will continue until the end of the received bits and at the end of the entire decoding process, the Viterbi Decoder will examine all possible paths and select one that has the minimum Hamming distance [8].

V. EXPERIMENTAL RESULTS

Figure 7 shows the implementation of the E2E system. The same model and HDL code are downloaded onto both FPGA based SDR and their connected computers contain the software interface model to be executed via Ethernet connection. The used AD 9364 FMCOMMS4-EBZ transceivers without using external amplifier has a maximum transmit power of 8 dBm and 2 dB noise figure on the receive side. Therefore, the E2E system is limited by the range where the two SDR are separated and thus, increasing separation leads to increasing degradation of the communication link [7].



Figure 7. Set up of E2E system sending and receiving messages using FDD

To quantify the E2E performance, the E2E design model includes error statistics by counting the number of error bits over the total received bits. This is the definition of Bits Error Rate (BER) and is done in the 'Configurable Software Interface' block of Figure 5. Equation 5 shows the equation for the BER specifically used for E2E system, where the division by 7 (represents the 7 bits per each decoded ASCII symbol) and 13 (13 Bernoulli random bits that were assembled along with the original message on the transmit side). In total, they are excluded from the BER calculation. Figure 8 shows the input from 'FPGA Receiver Logic' as complex 16-bit signed integers to the 'Reshape fcn', which reshapes the input complex values to 200 bits (174 message bits and 26 Barker header bits). The 'Descrambling' unpacked each payload to yield only the 174 message bits and the 'BitsToASCII' decodes the 174 message bits to ASCII characters [2].

$$BER = \left(\frac{\text{number error bits}}{\text{total received bits}} \right) \cdot \frac{1}{7} \cdot \frac{1}{13} \quad (5)$$

For BER determination, as incoming bits arrive, the accumulating error has been set to zero every 50 received frames so that the BER measurements are independent of each other. A comparison plot of the QPSK-1/2 and uncoded QPSK is shown in Figure 9. The plot shows the minimum BER versus the Line of Sight (LOS) separation distance between SDR-A and SDR-B with transmit-receive frequency at 915 MHz. The plot has been generated to take the minimum BER for each respective separation distance. For QPSK-1/2, the data shows BER = 0 for distances less than 14 m. In order to show the comparison on semi-log graph, BER = 0 is represented as the absolute minimum BER for the uncoded QPSK multiply by 10^{-3} . On the plot this is 3.45×10^{-10} and again in natural number it is 0.

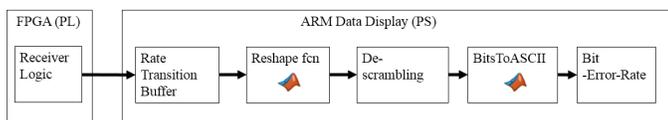


Figure 8. Detail blocks inside ARM Data Display (PS) executed by the software interface model

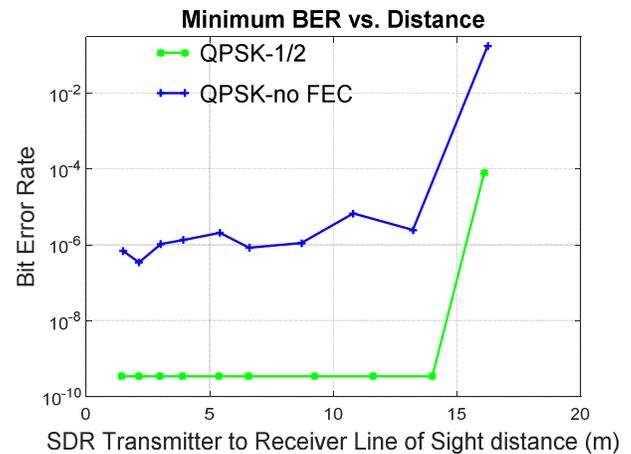


Figure 9. BER versus SDR transmitter to receiver LOS separation distance (SDR transmitter front-end is set to transmit at 915 MHz)

VI. CONCLUSION

This paper describes the design and implementation of an end to end digital wireless communication system or the E2E system on the FPGA from a model-based design system. Furthermore, the paper presented the design path to reconfigure the software controlled logic blocks for additional functionality and or complexity. The results show how to realize the E2E system and the BER vs. distance plot verified the theoretical expectation between uncoded QPSK and coded QPSK with FEC rate of 1/2 implemented over the air.

REFERENCES

- [1] "Ettus Research USRP N200/N210 Network Series," [Online]. Available: https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf. [Accessed 2017].
- [2] MathWorks, "Xilinx Zynq Based Radio Support from Communications System Toolbox," [Online]. Available: <https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/xilinx-zynq-based-radio-support-from-communications-system-toolbox.html>. [Accessed 2016].
- [3] AvNet, "Zynq™ Evaluation and Development - Hardware User's Guide," 27 January 2014. [Online]. Available: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf.
- [4] M. Rice, Digital Communications - A Discrete-Time Approach, Pearson, 2008.
- [5] M. Mukesh, L. Abhishek and R. R. Bhambare, "QPSK Modulator and Demodulator Using FPGA for SDR," *International Journal of Engineering Research and Applications*, vol. 4, no. 4, pp. 394-397, 2014.
- [6] M. Luke and R. Reggiannini, "Carrier frequency recovery in all-digital modems for burst-mode transmissions," *IEEE Transactions on Communications*, vol. 43, pp. 1169-1178, 1995.
- [7] Analog Devices, "RF Agile Transceiver AD9364 Data Sheet - Rev. C," 2014. [Online]. Available: <http://www.analog.com/media/en/technical-documentation/data-sheets/AD9364.pdf>.
- [8] "MIT 6.02," 6 October 2010. [Online]. Available: <http://web.mit.edu/6.02/www/f2010/handouts/lectures/L9.pdf>. [Accessed 1 April 2017].